QR-DECOMPOSITION BASED ALGORITHMS FOR ADAPTIVE VOLTERRA FILTERING

Mushtaq A. Syed and V. John Mathews

Department of Electrical Engineering
University of Utah
Salt Lake City, Utah 84112

Abstract - In this paper we present a pair of QR-RLS adaptive algorithms for second-order Volterra filtering. The algorithms are based solely on Given's rotations. Hence both the algorithms are numerically stable and highly amenable to parallel implementations. The computational complexity of one of the algorithms is comparable to that of the fast transversal Volterra filters. The algorithms can be easily extended to other types of polynomial nonlinearities.

I. INTRODUCTION

There are a number of applications in which the performance of linear filters is unacceptable and one has to resort to nonlinear filters. Nonlinear filters have been successfully used in several diverse applications [4]. In this paper we consider numerically stable algorithms for exponentially weighted recursive least squares (RLS) adaptive nonlinear filters equipped with a truncated Volterra series model [6, 7]. Early works on adaptive Volterra filters [1, 2, 3] were based on the LMS algorithm. Even though they are computationally simple, they suffer from slow and input signal-dependent convergence behavior and hence are not useful in many applications. Mathews and Lee [5] have presented a fast transversal algorithm for recursive least squares (RLS) adaptive Volterra filtering. The fast transversal Volterra filter is rapidly convergent; however, it suffers from poor numerical properties. Recently, the authors [9] presented computationally efficient and numerically stable RLS adaptive lattice algorithms for nonlinear filtering. In this paper we present another approach to the development of fast and numerically stable RLS algorithms for adaptive nonlinear filtering using QR-decomposition of the data matrix. The nonlinearity is modeled using a second-order Volterra series expansion; however,the results can be easily extended to other polynomial nonlinearities. We present two closely related algorithms for RLS second-order Volterra filtering. The computational complexity of one of the algorithms is comparable to that of fast RLS Volterra filters.

II. QR-RLS ADAPTIVE SECOND-ORDER VOLTERRA FILTER WITH BLOCK PROCESSING

We consider the problem of adaptively minimizing the exponentially weighted RLS cost function

$$\xi_N(n) = \sum_{k=0}^{n} \lambda^{n-k}(d(k) - \underline{W}^T(n)\underline{X}(k))^2 \qquad (1)$$

at each time, where λ is the weighting factor, $d(k)$ is the desired response signal, and $\hat{a}_{m_1}(n)$ and $\hat{b}_{m_1,m_2}(n)$ are the linear and quadratic coefficients, respectively, of the second-order Volterra filter. Define the input vector $\underline{X}(n)$ and the coefficient vector $\underline{W}(n)$, both of size $N(N+3)/2$ entries as

$$\underline{X}(n) = [x(n), x^2(n), x(n-1), \cdots, x(n), x(n-N+1)]^T \qquad (2)$$

and

$$\underline{W}(n) = [\hat{a}_1(n), \hat{b}_{1,1}(n), \hat{a}_2(n), \cdots, \hat{b}_{1,N-1}(n)]^T. \qquad (3)$$

We would like to develop computationally efficient and numerically stable algorithms to iteratively solve the optimization problem. This is achieved by transforming the nonlinear filtering problem into an equivalent multichannel linear filtering problem, illustrated by rewriting the entries of the input vector $\underline{X}(n)$ as

$$\begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-N+1) \\ x^2(n) & x^2(n-1) & \cdots & x^2(n-N+1) \\ & x(n)x(n-1) & \cdots & x(n-N+2)x(n-N+1) \\ & & \cdots & x(n-N+3)x(n-N+1) \\ & & \vdots & \vdots \\ & & \cdots & x(n)x(n-N+1) \end{bmatrix}. \qquad (4)$$

Each row of the above data matrix can be thought of as made up of samples of a signal belonging to a different channel. Note that the number of samples from each channel that are used in the estimation process is different. There are N samples from the first two channels that are used in the estimation process, $N-1$ samples from the third channel, $N-2$ from the fourth, and so on to the $(N+1)-th$ channel from which a single sample is used. There are $K = N+1$ channels. The signal at the $i-th$ channel is defined as $x_i(n) = x(n)x(n-i+1), i = 2, \cdots, N+1$ ($x_1(n) = x(n)$). $N_i = N-i+2, i = 3, \cdots, N+1$ ($N_1 = N_2 = N$) is the number of samples from the $i-th$ channel that are used in the estimation process and the total number of coefficients is

$$L = \sum_{i=1}^{N+1} N_i. \qquad (5)$$

Let us define the following $n \times 1$, error, desired, and input signal vectors:

$$\underline{e}(n) = [e(1), \cdots, e(n)]^T, \qquad (6)$$

$$\underline{d}(n) = [d(1), \cdots, d(n)]^T, \qquad (7)$$

and

$$\underline{x}_i(n) = [x_i(1), \cdots, x_i(n)]^T. \qquad (8)$$

The error vector $\underline{e}(n)$ in estimating $\underline{d}(n)$ is

0-7803-0593-0/92 $3.00 1992 IEEE

$$\underline{e}(n) = \underline{d}(n) - X_L(n)\underline{W}(n)\,, \tag{9}$$

where

$$X_L(n) = [\underline{x}_1(n)\cdots, \underline{x}_1(n-(N_1-N_2)), \underline{x}_2(n)\cdots \underline{x}_K(n-N_K+1)], \tag{10}$$

$$W(n) = [w_{1,1}(n), \cdots, w_{1,N_1-N_2}(n), \cdots, w_{K,N_K}(n)]^T, \tag{11}$$

and $(\cdot)^T$ denotes the transpose of $(\cdot)$. The adaptive filter tries to minimize the cost function which can be rewritten in matrix notation as

$$\xi(n) = \|B^{1/2}(n)\underline{e}(n)\|^2, \tag{12}$$

where $\|(\cdot)\|$ denotes the Euclidean norm of $(\cdot)$ and $B^{1/2}(n)$ is the $n \times n$ exponential weighting matrix

$$B^{1/2}(n) = diag[\sqrt{\lambda}^{n-1}, \cdots, \sqrt{\lambda}, 1]. \tag{13}$$

Let $Q_L(n)$ be the orthogonal matrix that triangularizes the weighted data matrix, *i.e.*,

$$Q_L(n)B^{1/2}(n)X_L(n) = \begin{bmatrix} R_L(n) \\ \underline{0} \end{bmatrix}, \tag{14}$$

where $\underline{0}$ denotes a zero matrix/vector of appropriate dimension, and $R_L(n)$ is an $L \times L$ upper triangular matrix. Let us also define the $L \times 1$ vectors $\underline{e}_u(n)$ and $\underline{U}(n)$ and $n-L \times 1$ vectors $\underline{e}_v(n)$ and $\underline{V}(n)$ as

$$Q_L(n)B^{1/2}(n)\underline{e}(n) = \begin{bmatrix} \underline{e}_u(n) \\ \underline{e}_v(n) \end{bmatrix} \tag{15}$$

and

$$Q_L(n)B^{1/2}(n)\underline{d}(n) = \begin{bmatrix} \underline{U}(n) \\ \underline{V}(n) \end{bmatrix}, \tag{16}$$

respectively. Then the LS coefficient vector $\underline{W}(n)$ can be computed by back substitution as

$$\underline{W}(n) = R_L^{-1}(n)\underline{U}(n) \tag{17}$$

and the estimation error can be computed using equation (9). It can be shown that $Q_L(n)$ can be recursively updated as

$$Q_L(n) = \hat{Q}_L(n)\begin{bmatrix} \tilde{Q}_L(n-1) & \underline{0} \\ \underline{0}^T & 1 \end{bmatrix}, \tag{18}$$

where $\hat{Q}_L(n)$ is the sequence of L rotations

$$\hat{Q}_L(n) = \hat{Q}_{(L)}(n)\cdots\hat{Q}_{(1)}(n) \tag{19}$$

such that

$$\hat{Q}_L(n)\begin{bmatrix} \sqrt{\lambda}R_L(n-1) \\ \underline{0} \\ \tilde{x}(n) \end{bmatrix} = \begin{bmatrix} R_L(n) \\ \underline{0} \\ \underline{0}^T \end{bmatrix} \tag{20}$$

and $\tilde{x}(n)$ is the last row of the data matrix $X_L(n)$. The first rotation $\hat{Q}_{(1)}(n)$ in equation (19) annihilates $x_1(n)$ by rotating it into $r_L(1,1)$ the element in the first row and first column of the upper triangular matrix $R_L(n)$. It can be constructed as

$$\hat{Q}_{(1)}(n) = \begin{bmatrix} I & 0 & 0 \\ cos\theta_1(n) & 0 & sin\theta_1(n) \\ 0 & I & 0 \\ -sin\theta_1(n) & 0 & cos\theta_1(n) \end{bmatrix} \tag{21}$$

where

$$cos\theta_1(n) = \sqrt{\lambda}r_L(1,1)/\sqrt{\lambda r_L^2(1,1) + x_1^2(n)} \tag{22}$$

and

$$sin\theta_1(n) = x_1(n)/\sqrt{\lambda r_L^2(1,1) + x_1^2(n)}. \tag{23}$$

The other rotations can be calculated in a similar fashion. $\underline{U}(n)$ and $\underline{V}(n)$ can be recursively updated using equation (18) as

$$\hat{Q}_L(n)\begin{bmatrix} \sqrt{\lambda}\underline{U}(n-1) \\ \sqrt{\lambda}\underline{V}(n-1) \\ d(n) \end{bmatrix} = \begin{bmatrix} \underline{U}(n) \\ \sqrt{\lambda}\underline{V}(n-1) \\ \alpha(n) \end{bmatrix}. \tag{24}$$

Define the pinning vector as $\underline{\sigma}(n) = [0, \cdots, 0, 1]^T$ and let

$$Q_L(n)\underline{\sigma}(n) = \hat{Q}_L(n)\underline{\sigma}(n) = \begin{bmatrix} \underline{a}_L(n) \\ \underline{0} \\ \gamma_L(n) \end{bmatrix}, \tag{25}$$

where $\gamma_L(n)$ is a scalar and $\underline{a}_L(n)$ is an $L \times 1$ vector. Using equations (25) and (24) one can show that the current estimation error can be calculated as

$$e(n) = \alpha(n)\gamma_L(n). \tag{26}$$

In QRD-based fast RLS algorithms, using the solutions to the forward and backward prediction problems, two equations are obtained for computing the orthogonal matrix $Q_{L+K}(n+1)$ that triangularizes $X_{L+K}(n+1)$. These two equations are used to solve for $Q_L(n+1)$, which is required at the next time step.

III. BACKWARD PREDICTION

In the L^{th} order backward prediction problem the matrix

$$\underline{d}_b(n) = [\underline{x}_1(n-N_1), \cdots, \underline{x}_K(n-N_K)] \tag{27}$$

is estimated using the data matrix $X_L(n)$. The $M \times K$ backward prediction error matrix $\underline{e}_b(n)$ is given by

$$\underline{e}_b(n) = \underline{d}_b(n) - X_L(n)\underline{W}_b(n), \tag{28}$$

where $\underline{W}_b(n)$ is the $L \times K$ backward prediction coefficient matrix. Similar to equation (24), we define

$$Q_L(n)B^{1/2}(n)\underline{d}_b(n) = \begin{bmatrix} \underline{U}_b(n) \\ \underline{V}_b(n) \end{bmatrix}. \tag{29}$$

Now consider triangularizing the augmented matrix $X_{L+K}(n)$. Equations (28) and (29) lead to

$$Q_L(n)B^{1/2}(n)X_{L+K}(n) = \begin{bmatrix} R_L(n) & \underline{U}_b(n) \\ \underline{0} & \underline{V}_b(n) \end{bmatrix}. \tag{30}$$

Let us define $Q_b(n)$ as the orthogonal matrix that rotates $\underline{V}_b(n)$ into an upper triangular matrix $\underline{\epsilon}_b(n)$ and completes the triangularization, as shown below.

$$Q_b(n)\begin{bmatrix} R_N(n) & \underline{U}_b(n) \\ \underline{0} & \underline{V}_b(n) \end{bmatrix} = \begin{bmatrix} R_L(n) & \underline{U}_b(n) \\ \underline{0} & \underline{\epsilon}_b(n) \\ 0 & 0 \end{bmatrix}. \tag{31}$$

Now consider triangularizing the augmented matrix $X_{L+K}(n+1)$ partitioned as

$$X_{L+K}(n+1) = \begin{bmatrix} X_L(n) & \underline{d}_b(n) \\ \underline{z}_1 & z_2 \end{bmatrix}, \tag{32}$$

where $\underline{z}_1$ and z_2 constitute the last row of the data matrix. $X_{L+K}(n+1)$ can be partially triangularized using equations (18), (30), and (31) as

$$\hat{Q}_L(n+1)\begin{bmatrix} Q_b(n) & \underline{0} \\ \underline{0} & 1 \end{bmatrix}\begin{bmatrix} Q_L(n) & \underline{0} \\ \underline{0}^T & 1 \end{bmatrix}\bar{B}^{1/2}(n+1)X_{L+K}(n+1) = \begin{bmatrix} R_L(n+1) & \underline{U}_b(n+1) \\ \underline{0} & \sqrt{\lambda}\underline{\epsilon}_b(n) \\ \underline{0} & 0 \\ \underline{0}^T & \underline{\alpha}_b(n+1) \end{bmatrix}. \tag{33}$$

Define $\hat{Q}_b(n+1)$ such that

$$\hat{Q}_b(n+1)\begin{bmatrix} R_L(n+1) & \mathrm{U}_b(n+1) \\ 0 & \sqrt{\lambda}\underline{\epsilon}_b(n) \\ 0 & 0 \\ 0^T & \underline{\alpha}_b(n+1) \end{bmatrix} = \begin{bmatrix} R_{L+K}(n+1) \\ 0 \end{bmatrix}. \quad (34)$$

Equations (33) and (34) lead to

$$Q_{L+K}(n+1) = \hat{Q}_b(n+1)\hat{Q}_L(n+1)\begin{bmatrix} Q_b(n) & 0 \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} Q_L(n) & 0 \\ 0^T & 1 \end{bmatrix}. \quad (35)$$

IV. FORWARD PREDICTION

In the L^{th} order forward prediction problem the matrix

$$\mathrm{d}_f(n) = [\mathrm{x}_1(n), \cdots, \mathrm{x}_K(n)] \quad (36)$$

is estimated using the data matrix $X_L(n-1)$. The $M \times K$ forward prediction error matrix $\underline{\mathrm{e}}_f(n)$ is given by

$$\mathrm{e}_f(n) = \mathrm{d}_f(n) - X_L(n-1)\mathrm{W}_f(n), \quad (37)$$

where $\mathrm{W}_f(n)$ is the $L \times K$ forward coefficient matrix. Similar to equation (30) we define

$$Q_L(n-1)B^{1/2}(n-1)\mathrm{d}_f(n) = \begin{bmatrix} \mathrm{U}_f(n) \\ \mathrm{V}_f(n) \end{bmatrix}. \quad (38)$$

Now consider the triangularization of $X_{L+K}(n+1)$ partitioned as

$$X_{L+K}(n+1) = \begin{bmatrix} X_L(n-1) & [\mathrm{x}_1(n), \cdots, \mathrm{x}_K(n)] \\ \mathrm{s}_1 & \mathrm{s}_2 \end{bmatrix} P, \quad (39)$$

where the vectors s_1 and s_2 constitute the last row of the matrix $X_{L+K}(n+1)$ after its columns have been permuted by an appropriate permutation matrix P. Operating with $Q_L(n-1)$ gives

$$\begin{bmatrix} 1 & 0^T & 0 \\ 0 & Q_L(n-1) & 0 \\ 0 & 0^T & 1 \end{bmatrix} B^{1/2}(n+1)X_{L+K}(n+1) = \begin{bmatrix} 0^T & \hat{x} \\ \sqrt{\lambda}R_L(n-1) & \sqrt{\lambda}\mathrm{U}_f(n) \\ 0 & \sqrt{\lambda}\mathrm{V}_f(n) \\ \mathrm{s}_1 & \mathrm{s}_2 \end{bmatrix} P, \quad (40)$$

where $\hat{x} = \sqrt{\lambda}[x_1(1), \cdots, x_K(1)]$. Now, define $Q_f^v(n)$ such that

$$\begin{bmatrix} Q_f^v(n) & 0 \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} 0^T & \hat{x} \\ \sqrt{\lambda}R_L(n-1) & \sqrt{\lambda}\mathrm{U}_f(n) \\ 0 & \sqrt{\lambda}\mathrm{V}_f(n) \\ \mathrm{s}_1 & \mathrm{s}_2 \end{bmatrix} P = \begin{bmatrix} 0^T & 0^T \\ \sqrt{\lambda}R_L(n-1) & \sqrt{\lambda}\mathrm{U}_f(n) \\ 0 & \sqrt{\lambda}\underline{\epsilon}_f(n) \\ 0 & 0 \\ \mathrm{s}_1 & \mathrm{s}_2 \end{bmatrix} P, \quad (41)$$

where $\underline{\epsilon}_f(n)$ is a $K \times K$ element upper triangular matrix. The vector s_1 can be annihilated by $\hat{Q}_L(n)$.

$$\begin{bmatrix} 1 & 0^T \\ 0 & \hat{Q}_L(n) \end{bmatrix}\begin{bmatrix} 0^T & 0^T \\ \sqrt{\lambda}R_L(n-1) & \sqrt{\lambda}\bar{U}_f(n) \\ 0 & \sqrt{\lambda}\underline{\epsilon}_f(n) \\ 0 & 0 \\ \mathrm{s}_1 & \mathrm{s}_2 \end{bmatrix} P = \begin{bmatrix} 0^T & 0^T \\ R_L(n) & \mathrm{U}_f(n+1) \\ 0 & \sqrt{\lambda}\underline{\epsilon}_f(n) \\ 0 & 0 \\ 0^T & \underline{\alpha}_f(n+1) \end{bmatrix} P. \quad (42)$$

Define $\hat{Q}_f^v(n+1)$ such that

$$\hat{Q}_f^v(n+1)\begin{bmatrix} 0^T & 0^T \\ R_L(n) & \mathrm{U}_f(n+1) \\ 0 & \sqrt{\lambda}\underline{\epsilon}_f(n) \\ 0 & 0 \\ 0^T & \underline{\alpha}_f(n+1) \end{bmatrix} P = \begin{bmatrix} 0^T & 0^T \\ R_L(n) & \mathrm{U}_f(n+1) \\ 0 & \underline{\epsilon}_f(n+1) \\ 0 & 0 \\ 0^T & 0^T \end{bmatrix} P. \quad (43)$$

Now, if we operate with the permutation matrix P, then the matrix on the right-hand side of the above equation will no longer be triangular as P will be permuting the columns of the triangular matrix. Define $Q_f^u(n)$ as the rotation matrix that completes the triangularization and J as the shift matrix such that

$$JQ_f^u(n+1)\begin{bmatrix} 0^T & 0^T \\ R_L(n) & \bar{U}_f(n+1) \\ 0 & \underline{\epsilon}_f(n+1) \\ 0 & 0 \end{bmatrix} P = \begin{bmatrix} R_{L+K}(n+1) \\ 0 \end{bmatrix}. \quad (44)$$

It should be noted that the computation of $Q_f^u(n+1)$ does not require the triangular matrix $R_L(n)$. Equations (40)-(44) lead to

$$Q_{L+K}(n+1) = JQ_f^u(n+1)\hat{Q}_f^v(n+1)\begin{bmatrix} 1 & 0^T \\ 0 & \hat{Q}_L(n) \end{bmatrix}\begin{bmatrix} Q_f^v(n) & 0 \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} 1 & 0^T & 0 \\ 0 & Q_L(n-1) & 0 \\ 0 & 0^T & 1 \end{bmatrix}. \quad (45)$$

V. ALGORITHM

A computationally efficient algorithm can be obtained by operating on the pinning vector $\underline{\sigma}(n+1)$ with $Q_{L+K}(n+1)$ and using equations (35) and (45). Equation (35) gives

$$\hat{Q}_b(n+1)\begin{bmatrix} \mathrm{a}_L(n+1) \\ 0 \\ \gamma_L(n+1) \end{bmatrix} = \begin{bmatrix} \mathrm{a}_{L+K}(n+1) \\ 0 \\ \gamma_{L+K}(n+1) \end{bmatrix}. \quad (46)$$

From equation (34) it is clear that, since $\hat{Q}_b(n+1)$ rotates the K zeros below $\mathrm{a}_L(n+1)$ with $\gamma_L(n+1)$to generate $\mathrm{a}_{L+K}(n+1)$, the top L components of $\underline{\mathrm{a}}_{L+K}(n+1)$ constitute $\underline{\mathrm{a}}_L(n+1)$. Using equation (45) and again operating on $\underline{\sigma}(n+1)$, leads to

$$JQ_f^u(n+1)\hat{Q}_f^v(n+1)\begin{bmatrix} 0 \\ \mathrm{a}_L(n) \\ 0 \\ \gamma_L(n) \end{bmatrix} = \begin{bmatrix} \mathrm{a}_{L+K}(n+1) \\ 0 \\ \gamma_{L+K}(n+1) \end{bmatrix}. \quad (47)$$

Now, $\hat{Q}_f^v(n+1)$ rotates $\gamma_L(n)$ into the K zeros below the vector $\underline{\mathrm{a}}_L(n)$ to generate the $K \times 1$ vector $\underline{\mathrm{h}}(n)$ and in the process $\gamma_L(n)$ is transformed into $\gamma_{L+K}(n+1)$.

$$Q_f^v(n+1)\begin{bmatrix} 0 \\ \mathrm{a}_L(n) \\ 0 \\ \gamma_L(n) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathrm{a}_L(n) \\ \mathrm{h}(n) \\ 0 \\ \gamma_{L+K}(n+1) \end{bmatrix}. \quad (48)$$

Finally, $Q_f^u(n+1)$ rotates $\underline{\mathrm{h}}(n)$ into $\underline{\mathrm{a}}_L(n)$ to form $\underline{\mathrm{a}}_{L+K}(n+1)$.

$$\begin{bmatrix} \mathrm{a}_{L+K}(n+1) \\ 0 \\ \gamma_{L+K}(n+1) \end{bmatrix} = JQ_f^u(n+1)\begin{bmatrix} 0 \\ \mathrm{a}_L(n) \\ \mathrm{h}(n) \\ 0 \\ \gamma_{L+K}(n+1) \end{bmatrix}. \quad (49)$$

We now have all the equations for computing $\hat{Q}_L(n+1)$ efficiently. The algorithm for QR-RLS second-order Volterra filtering is given in Table 1. The overall computational complexity is $O(N^4)$ arithmetic operations per time step.

VI. QR-RLS ADAPTIVE SECOND-ORDER VOLTERRA FILTER WITH SEQUENTIAL PROCESSING

The block processing algorithm presented in the previous section processes all the $N+1$ channels simultaneously. In this section we will present another algorithm that processes the channels sequentially. The sequential algorithm is computationally more efficient than the block algorithm because unlike the block algo-

rithm it does not require any matrix processing. The computational complexity of the sequential algorithm is $O(N^3)$ arithmetic operations per time step which is comparable to that of other fast RLS adaptive Volterra filters. Since the channels are processed individually, this corresponds exactly to processing in the single-channel algorithm. Hence knowledge of the single-channel algorithm can be carried over to the multichannel algorithm. Also, sequential processing leads to regularity in implementation which results in a modular architecture [8]. Since the derivation of the algorithm is very similar to the derivation of the block algorithm, the details are omitted. The algorithm is given in Table 2.

VII. CONCLUSIONS

In this paper we presented a pair of QR-RLS adaptive algorithms for second-order Volterra filtering. Both the algorithms are based solely on Given's rotation. Hence both are numerically stable and highly amenable to parallel implementations using arrays. One of the algorithms is a block processing algorithm in the sense that it processes all the channels simultaneously. The other processes the channels sequentially. The sequential algorithm is computationally much more efficient than the block algorithm and is comparable to that of fast RLS Volterra filters. Another attractive feature of sequential processing is that knowledge of the single-channel algorithm can be applied to the multichannel case. We have carried out extensive experiments to study the performances of the two algorithms in finite precision. The results indicate that the performances of the two algorithms are nearly the same. Also, both the algorithms appear to be numerically stable and retain the fast convergence property of RLS algorithms.

ACKNOWLEDGMENT

This work was supported in part by the NSF under Grant MIP-8922146.

References

[1] M. J. Coker and D. N. Simkins, "A nonlinear adaptive noise canceller," *Proc. ICASSP,* pp. 470-473, 1980.

[2] T. Koh and E. J. Powers, "An adaptive nonlinear digital filter with lattice orthogonalization," *Proc. ICASSP,* Boston, pp. 37-40, April 1983.

[3] T. Koh and E. J. Powers, "Second order Volterra filtering and its applications to nonlinear system identification," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* Vol. ASSP-33, No. 6, pp. 1445-1455, December 1985.

[4] V. J. Mathews, "Adpative Polynomial Filters," *IEEE Signal Processing Magazine,* pp. 10-26, July 1991.

[5] V. J. Mathews and J. Lee, "A fast recursive least-squares second order Volterra filter," *Proc. ICASSP,* New York, pp. 1383-1386, April 1988.

[6] W. J. Rugh, *Nonlinear System Theory,* Baltimore: Johns Hopkins University Press, 1981.

[7] M. Schetzen, *The Volterra and Wiener Theory of the Nonlinear Systems,* New York: John Wiley, 1980.

[8] D. T. M. Slock, *et al.,* "Modular and Numerically Stable Multichannel FTF Algorithms," *Proc. ICASSP,* pp. 1039-1041, Glasgow, May 1989.

[9] M. A. Syed and V. J. Mathews, "Lattice and QR Decomposition-Based Algorithms for Recursive Least Squares Adaptive Nonlinear Filters," *Proc. ISCAS,* New Orleans, pp. 262-265, May 1990.

Table 1: QR-RLS adaptive second-order Volterra algorithm with block processing.

Given $d(n)$, $x(n)$, $\underline{\mathrm{U}}_f(n-1)$, $\underline{\epsilon}_f(n-1)$, $\underline{\mathrm{a}}_L(n-1)$, $\gamma_L(n-1)$, and $\hat{Q}_L(n-1)$.

DO equations (1.1) – (1.10) for $n = 1$ onwards.

$x_1(n) = x(n)$, $x_2(n) = x^2(n)$

$x_j(n) = x(n)x(n-j+2)$, $j = 3, \cdots, N+1$ (1.1)

$$\hat{Q}_L(n-1)\begin{bmatrix} \sqrt{\lambda}\underline{\mathrm{U}}_f(n-1) \\ x_1(n), \cdots, x_{N+1}(n) \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{U}}_f(n) \\ \underline{\alpha}_f(n) \end{bmatrix} \quad (1.2)$$

$$\hat{Q}_f^v(n)\begin{bmatrix} \sqrt{\lambda}\underline{\epsilon}_f(n-1) \\ \underline{\alpha}_f(n) \end{bmatrix} = \begin{bmatrix} \underline{\epsilon}_f(n) \\ 0 \end{bmatrix} \quad (1.3)$$

$$\hat{Q}_f^v(n) = \begin{bmatrix} 0 \\ \gamma_L(n-1) \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{h}}(n) \\ \gamma_{L+N+1}(n) \end{bmatrix} \quad (1.4)$$

$$Q_f^v(n)\begin{bmatrix} \underline{\mathrm{U}}_f(n) \\ \underline{\epsilon}_f(n) \end{bmatrix} P = \begin{bmatrix} * \\ 0 \end{bmatrix} \quad (1.5)$$

$$Q_f^v(n)\begin{bmatrix} \underline{\mathrm{a}}_L(n-1) \\ \underline{\mathrm{h}}(n) \end{bmatrix} = \underline{\mathrm{a}}_{L+N+1}(n) \quad (1.6)$$

$$\gamma_L(n) = +\sqrt{1 - \|\underline{\mathrm{a}}_L(n)\|^2} \quad (1.7)$$

$$\hat{Q}_L(n)\underline{\alpha}(n) = \begin{bmatrix} \underline{\mathrm{a}}_L(n) \\ \gamma_L(n) \end{bmatrix} \quad (1.8)$$

$$\hat{Q}_L(n)\begin{bmatrix} \sqrt{\lambda}\underline{\mathrm{U}}(n-1) \\ d(n) \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{U}}(n) \\ \alpha(n) \end{bmatrix} \quad (1.9)$$

$$e(n) = \alpha(n)\gamma_L(n) \quad (1.10)$$

Table 2: QR-RLS adaptive second-order Volterra algorithm with sequential processing

Given $d(n)$, $x(n)$, $\underline{\mathrm{U}}_i^f(n-1)$, $\underline{\epsilon}_{i,L}^f(n-1)$, $\underline{\mathrm{a}}_{i-1,L}(n-1)$, $\gamma_{i-1,L}(n)$ and $\hat{Q}_{i-1,L}(n)$.

DO equations (2.1) to (2.10) for $n = 1$, *onwards.*

$x_1(n) = x(n)$, $x_2(n) = x^2(n)$

$x_j(n) = x(n)x(n-j+2)$, $j = 3, \cdots, N+1$ (2.1)

DO equations (2.2) to (2.7) for $i = 1, \cdots, N+1$.

$$Q_{i-1,L}(n)\begin{bmatrix} \sqrt{\lambda}\underline{\mathrm{U}}_i^f(n-1) \\ x_i(n) \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{U}}_i^f(n) \\ \alpha_{i,L}^f(n) \end{bmatrix} \quad (2.2)$$

$$\hat{Q}_{i,L}^{v\,f}(n)\begin{bmatrix} \sqrt{\lambda}\epsilon_{i,L}^f(n-1) \\ \alpha_{i,L}^f(n) \end{bmatrix} = \begin{bmatrix} \epsilon_{i,L}^f(n) \\ \underline{0}^T \end{bmatrix} \quad (2.3)$$

$$\hat{Q}_{i,L}^{v\,f}(n)\begin{bmatrix} \underline{0} \\ \gamma_{i-1,L}(n) \end{bmatrix} = \begin{bmatrix} h_i(n) \\ \gamma_{i,L+1}(n) \end{bmatrix} \quad (2.4)$$

$$Q_{i,L}^{u\,f}(n)\begin{bmatrix} \underline{\mathrm{U}}_{i,L}^f \\ \epsilon_{i,L}^f(n) \end{bmatrix} = \begin{bmatrix} * \\ \underline{0} \end{bmatrix}, \; * \equiv \text{'don't care'} \quad (2.5)$$

$$Q_{i,L}^{u\,f}(n)\begin{bmatrix} \underline{\mathrm{a}}_{i-1\,L}(n) \\ h_i(n) \end{bmatrix} = \underline{\mathrm{a}}_{i,L+1}(n) \quad (2.6)$$

$$\gamma_{i,L}(n) = \sqrt{1 - \|\underline{\mathrm{a}}_{i,L}(n)\|^2} \quad (2.7)$$

$$\hat{Q}_{i,L}(n)\underline{\alpha} = \begin{bmatrix} \underline{\mathrm{a}}_{i,L}(n) \\ \gamma_{i,L}(n) \end{bmatrix} \quad (2.8)$$

$$\hat{Q}_{N+1,L}(n)\begin{bmatrix} \sqrt{\lambda}\underline{\mathrm{U}}(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{U}}(n) \\ \alpha(n) \end{bmatrix} \quad (2.9)$$

$$e(n) = \gamma_{N+1,L}(n)\alpha(n) \quad (2.10)$$